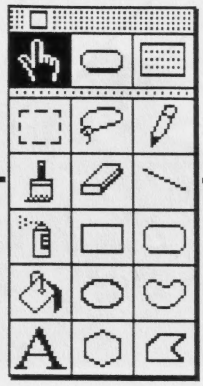
The Apple HyperCard User Group — Issue #5-December 3, 1987

Windoid



A Publication for the Informed HyperCard User

In this fifth issue of **WINDOID,** Sioux Lacy gives you important information about HyperCard external commands (XCMDs) and external functions (XFCNs). These are the "Magic Hooks" that can be written in "C", Pascal, or 68000 assembly language and used to create exciting extensions to HyperCard functionality. Additionally, Phil Wyman has given us more HyperCard User Tips, Robin Shank explains some HyperCard features, James Redfern gives us an explanation of some HyperTalk concepts, and Paul Foraker teaches us how to show hidden buttons and fields and set scripts of objects on the fly.

I have had many requests for issues of WINDOID. If you would like a copy, or copies, please send a self-addressed stamped envelope (SASE) to the address listed on the form at the back of each issue. Thank you for your continued interest in WINDOID.

In This Issue

- About XCMDs & XFCNs
- HyperCard User Tips
- It's a Feature
- Some Things Useful
- Wanted: Whole Earth Ad
- Showing & Setting

About XCMDs & XFCNs...

By Sioux Lacy

Tips & Techniques for Users of External Code Resources

Many of you may not be aware of yet another tool HyperCard has provided for the stack designer / user. External commands or functions can be used to create an extension to HyperCard functionality. Accessed much in the same manner that user-defined commands and functions are, "XCMD"s and "XFCN"s are code resources that can be attached to any stack.

If you have a bug, suggestion, comment, or just want to know the best way to do something in HyperCard, you can fill out the form and send it to:

AHUG c/o David Leffler
Apple Computer, Inc.
MS/27-AQ
20525 Mariani Ave.
Cupertino, CA 95014

Or copy the format and Apple-Link™ it to:

HYPERBUG$

Several external commands are included with the HyperCard product: "Flash" is an XCMD that will flash the screen (invert black and white, then invert back again). It's attached to the HyperCard application so it can be accessed from any stack with the command "Flash", followed by the number of times you would like the screen to flash.

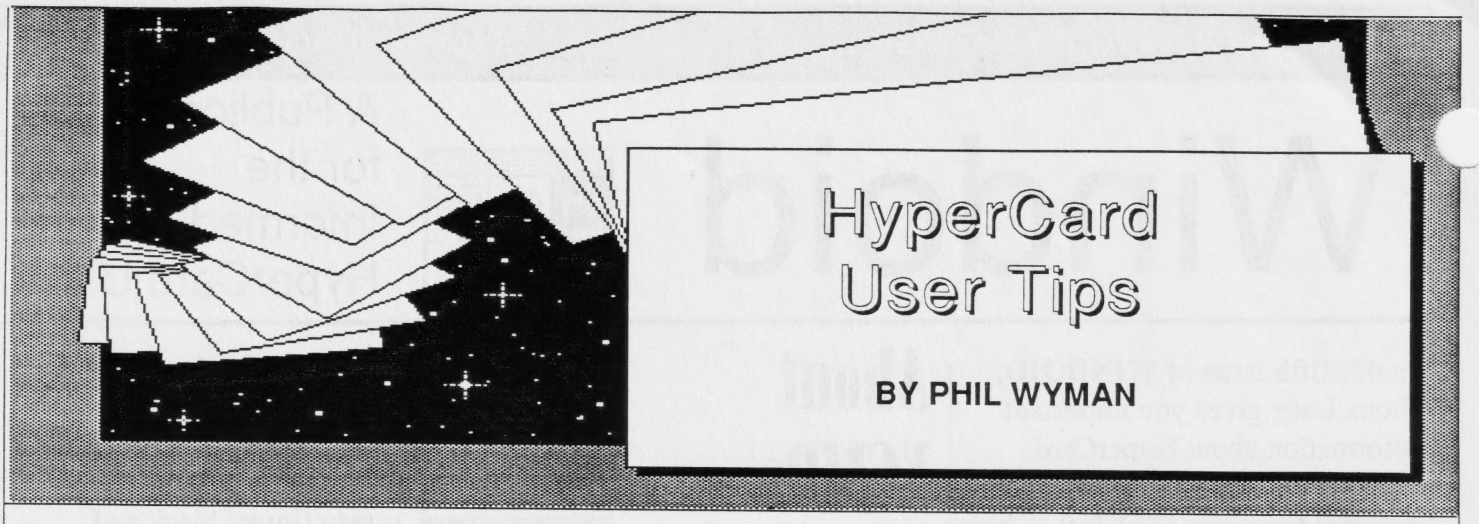
If you look in the Documents Representative stack, you will find another example of an external command: "Getdocs". It is called by the "Update" button's script. A line in the "mouseUp" handler reads:

```
getdocs line i of documents
```

This is a call to the "Getdocs" command, passing it one argument which is a folder name from the Documents card in your Home stack. "Getdocs" responds by creating the cards in the Documents stack.

Already some interesting externals commands have appeared as shareware. Some of them provide the user with a button that will attach the external code to your stack, but if you discover one that hasn't, you can use the following method to do so:

continued on page 4



HyperCard User Tips

BY PHIL WYMAN

1. There is a form of the Repeat command which works with "down to". "Down to" is two words in this syntax. This form of the repeat command will iterate your variable downward each step through the loop.

```
repeat with x = 100 down to 1
   put x into field 1
   go next card
end repeat
```

An example of a useful application of the "repeat down to" command is if you're deleting all the buttons or deleting all the fields of a card or background. For instance,

```
put the number of buttons of this
card into var
repeat with x = var down to 1
   choose button tool
   click at the loc of button x
   doMenu "Cut Button"
end repeat
```

If you had done this with "x = 1 to var" instead "x = var down to 1" then soon you would be clicking at a button that didn't exist.

2. There's a nice debugging feature in the Home Stack Script. The handler is called searchscript, and its two parameters are first the string you are searching for, and second the stack that you are going to be searching for that string. What if, for example, you had hundreds of cards with scripts, and you found you misspelled a word in one script. You could then search all your scripts to find which ones had the same misspelling error. You would say in the message box:

```
searchscript "mispelled word","stackname"
```

3. You do not have to have your cursor at the end of your message in the message window in order to execute your message. In the message window, you may be typing along on the fifth word and found that you've made a typing error on the first word. You click on the first word and correct your error. Now the cursor is in the first word. At this point you can still hit the return or enter keys to execute the message window, even though your cursor is not at the end of the message.

4. An easy way to copy a current background in your stack into a new background in the same stack is:

Copy card, go to a temporary stack, paste Card. Immediately copy card, go to original stack and paste the card. This should create a new background in your stack that is exactly like your old one.

If you need to copy the foreground of a card to the current background, try the free utility stack "Foregnd to Bkgnd".

"IT'S A FEATURE"

Explanations of some HyperCard features

by Robin Shank

Drawing Through Scripts:

Because the Polygon tool is controlled differently than the other drawing tools, the Drag From command won't work in this tool. If you need to plot irregular shapes, use the Curve tool or the Line tool instead. The Lasso tool depends on an enclosing movement to select something, but a script will only drag in a straight line.

In many cases when you draw through a script, you will want to set the dragSpeed down to around 500. With the dragSpeed at the default setting, the Spray tool will only show the starting and ending points, because the spray cycle is slower than the movement on the screen. Also, with Draw Multiple on, you won't see any multiples unless the dragSpeed is slowed down.

Printing Notes:

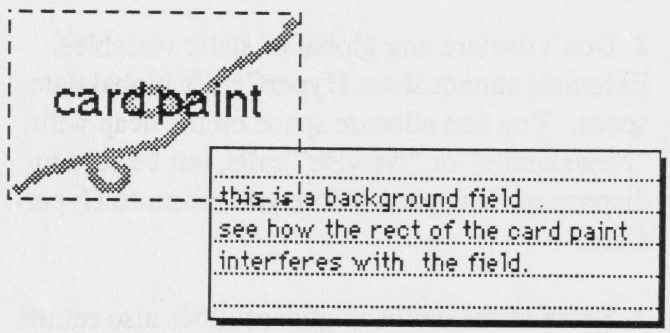
Background fields that are overlapped by another object or card picture will print at 72 dpi on the LaserWriter. A piece of card paint whose *rect* overlaps a background field will cause that background field to print at 72 dpi. (The rect can be shown by the rectangle formed by the marching ants when selected by the marquis.) One exception: if a transparent button that does not show a name or icon covers a background field, that field will still print at high resolution.

After printing a stack or report, the format is saved as the default. However, when printing cards through a script using Open printing with dialog, the dialog will show the default format determined by the last Print Stack, but changes to the printing format are not saved and affect only that one print job.

MultiFinder...

750K is the absolute minimum memory configuration recommended to run HyperCard under MultiFinder. If you run HC under MultiFinder on a 1 Mb machine, you will be limited to Typing UserLevel, and processes that need more memory (such as pasting graphics or printing) will, at the least complain, and at the worst, exit grumpily back to the Finder (who said crash??).

When printing under MultiFinder with Background Printing ON, Print Stack sends copious amounts of data to the LaserWriter in a very short period of time. If a spooler of any kind is catching that before sending it to the printer, it is likely to fill up very quickly. A MultiFinder Spool folder for a 10-card, 20K stack could easily be over 1 Mb. (500K+ per card is being sent to the LaserWriter.)



To attach an existing external command to one of your stacks, you will need to use a resource mover, like ResEdit, to copy it from its current stack and paste it into your stack. In ResEdit, this procedure works as follows:
1) Launch ResEdit.
2) Find the stack with the external command or function that you want to copy, and open that stack.
3) You will see a resource type of XCMD (or XFCN).
4) Select and copy the resource.
5) Then find and open your stack.
6) It will probably ask you if you want to open a "resource fork" for your stack (unless it already has one).
7) Click "OK", and ResEdit will open a window into which you can paste the resource.

About XCMDs...

Tips & Techniques for Writers of External Code Resources

For you Pascal, C, or 68000 assembly language programmers who want to write your own external commands, here are a couple of hints:

1 Most importantly, do get the "HyperCard Developer's Toolkit" package from APDA[1]. This package includes a disk, called "Magic Hooks", with sample source files in Pascal and C, and documentation.

2 Don't declare any global or static variables. Externals cannot share HyperCard's global data space. You can allocate space on the heap with "NewHandle" or "NewPtr" calls, but be sure to dispose of that space before you return to HyperCard.

3 Some of the callback glue routines also return handles to the caller (specifically, PasToZero, EvalExpr, GetGlobal, GetFieldByName, GetFieldByNum, and GetFieldByID). In these cases, HyperCard has allocated the space associated with these handles, and your code is responsible for deallocating it once you've examined its contents.

4 The rule about disposing handles and pointers that your code allocates has one exception: if you allocate space in order to return a string to HyperCard in the "returnValue" element of the XCMD parameter block. HyperCard has not pre-allocated any heapspace for that handle. You must do the allocation in your code, then copy your return message into that allocated space. HyperCard will dispose of the space for you.

An example that combines usage of a callback routine and assignment to a returnValue follows. Note that in this case, you do not deallocate the handle. This example uses the "PasToZero" callback routine to copy a string and zero-terminate it for return to HyperCard.

```
paramPtr^.returnValue := PasToZero('An
error has occurred.');
```

5 For testing externals, one useful technique is to call your code repeatedly from HyperCard and "put the heapspace" in between the calls. If you see a rapidly diminishing heap, you might suspect that your external is allocating space and forgetting to dispose of it.

6 An error in the "Magic Hooks" documentation: the callback routines, "ExtToStr" and "StrToExt" convert floating point numbers to strings and vice versa — they do not convert extended long integers. Oops, sorry about the error.

7 HyperCard stores strings as zero-terminated arrays of any length. This is consistent with the way 'C' handles strings, but not Pascal. So, for example, when a Pascal code resource receives parameters, it will need to convert the strings to Pascal strings (beginning with a length byte, and no zero-terminator). Read the documentation for the callback glue routines carefully. Some of those routines return zero-terminated strings, and others return Pascal strings. Your code will need to convert, if appropriate.

continued on page 5

Also, remember that a Pascal string is limited in length to 255 characters. So, for example, if you have a zero-terminated string that is longer than that, and you call ZeroToPas to convert it for you, HyperCard will truncate your string to 255 characters.

8 If you're writing external code in MPW 'C', there are some additional constraints that you need to be aware of regarding string literals. In an environment that has global and static data initialization, you can copy a string to the space referenced by a handle in this manner:

```
myHndl = NewHandle( STRINGSIZE );
strcpy( *myHndl, "Hello world" );        /* probably should lock this handle */
                                         /* and unlock it after the strcpy call */
```

In an MPW C[2] external code resource, you cannot use a string literal. Instead, you must create an STR# resource that contains your strings (use ResEdit to do this), and then retrieve them with a call to GetIndString:

```
char myString[ STRINGSIZE ];             /* make an array large enough */

GetIndString( myString, RESOURCE_ID, STRING_NUM );
```

9 It is not necessary to initialize the various Tool Managers in your external code. HyperCard has already done this, and in some cases, such as the GrafPort, reinitialization can be hazardous.

1 APDA, Apple Programmer's & Developer's Association is located at:
290 SW 43rd Street
Renton, WA 98055

2 LightSpeed C will allow the string literal to be passed as an argument.

SOME THINGS USEFUL

JAMES REDFERN

This is a short explanation of some concepts that are useful in using HyperTalk:

• A container is what you put your information into. Containers are: "it", "selection", the message box, a field, or a variable. Containers can be thought of as open or closed. An open container displays information on the screen; the contents of a closed container cannot be viewed. Variables and "it" are closed containers. The number of characters in an open container must be less than 32K, probably less than 30K to be safe, and less than 5K if a scrolling field is to respond quickly. The number of characters in a closed container are limited by available memory. (An exception is the use of the prepositions "before" or "after" with a closed container, which limits the number of characters to 64K.)

• Information is represented as a string of characters. Strings can sometimes be further interpreted as numbers or the logical values: "true" and "false". A number can be additionally interpreted as an integer or unsigned integer.

• When HyperCard performs an operation that requires numbers as operators, such as addition, it will try to interpret, i.e., convert, the strings as numbers. If the resulting number is put into a closed container, the container will retain the full precision of the internal representation. If the resulting number is put into an open container, the number must be

continued on page 8

WANTED:

StackWare Reviews and Recommendations for Whole Earth Ad

The Whole Earth Catalog rides again! Apple and the Point Foundation have collaborated on a HyperCard version of the Whole Earth Catalog, which was shown at MacWorld and gained significant media attention. Now we are beginning phase two of this project, which will incorporate reviews and recommendations of StackWare and HyperCard tools and accessories. Because of the rapid proliferation of stacks, we must rely on you, the users, to bring significant developments to our attention. Particularly, we are looking for recommendations or short reviews of:

• HyperCard Tools: These can take the form of stacks incorporating significant extensions to HyperCard, in the form of scripts or XCMDs. Examples might be online service front-ends, techniques for scrolling links, resource moving and conversion utilities, generalized text import or export techniques. Non-stack tools might include: useful DAs or companion programs to assist in building stacks, data import devices (scanners, digitizers, ...) which help convert material into stack format, and books, magazines, and teaching aids which help in learning or using HyperCard.

• Stacks with significant content: HyperCard was designed as an information vehicle for the non-programmer. Many significant stacks may not show a lot of programming "flash" but will contain vital data. For example, star catalogs, pharmaceutical directories, industry guides, and so on. Let us know what's out there.

• Stackware as a new media artform: We are beginning to see stacks that use HyperCard as a fusion artform, collaging graphics, sound, stored dynamics, and browsing into an interactive experience. We are interested in stacks leading in this direction of "information as art."

• Things we've missed: This area is moving so fast that we have undoubtedly overlooked major categories and themes. Please bring them to our attention.

Keep in mind that we are in the early days of stackware development. The items we include should have "staying power"—they should be of such a quality that they will still be of interest in a year, let's say. We encourage electronic submission of reviews—please include the stack or tool itself if it is freeware or shareware, otherwise please give us the full reference to the publisher so that we may verify pricing and availability information.

All submissions will be considered, though not all will be used. Submissions used will earn you a byline in the first-ever electronic Whole Earth Catalog and the knowledge that you have done a Good Deed for the HyperCard user community. Submissions may also be considered for inclusion in the Whole Earth Review magazine, which carries a monetary reward as well.

Here are our addresses:

Uucp: ucbvax!well!hank
Compuserve: leave message in WEC sig
Applelink: OREN1
SnailMail:
Whole Earth Stackware Toolbox
27 Gate Five Rd.
Sausalito, CA 94965

Thanks!

Showing and Setting

by
Paul Foraker

Lost and Found

In the process of designing your own stacks, once in a while you're likely to forget things. If you've ever hidden a field or a button from yourself, you might appreciate this little script.

--make a new button anywhere on your card
--(you'll delete it later) & put this script in it:

```
on mouseUp
  repeat with i = 1 to the number of fields
    --or bkgnd buttons, or card fields, or buttons
  show field i
    --or bkgnd button i, etc.
  end repeat
end mouseUp
```

Now, when you click on this temporary button, all your hidden fields or buttons will come into view.

Setting Scripts of Objects on the Fly

There are times when you want a HyperTalk script to make new objects (like buttons or fields) and put scripts into them. The command for getting scripts into objects is **set script of <object> to <source>.** This command can be used in a number of different ways. Let's look at some examples (for the sake of simplicity, I've assumed that we're using a simple button mouseUp handler to set the script of a button named, "target"):

--getting the new script directly from the executing handler
--(remember not to use Option-Return *inside* a quoted string)

```
on mouseUp
  set script of button "target" to "on mouseUp" &
  return & "go next card" & return & "end mouseUp" & return
end mouseUp
```

--getting the new script from a field

```
on mouseUp
  set script of button "target" to field 1
end mouseUp
```

--getting the new script from another object

```
on mouseUp
  set script of button "target" to script of button 2
end mouseUp
```

--modifying an existing script
--let's say we think the old script was simply (on mouseUp) "go next
--card", but we're not sure; it might already have a visual effect,
--and we want to add a visual effect:

```
on mouseUp
    --first check for a visual effect and get out if there is one
  if script of button "target" contains "visual" then exit mouseUp
    --if not, then modify the script
  put script of button "target" into temp
  put return & "visual effect wipe left" after
  line 1 of temp
  set script of button "target" to temp
end mouseUp
```

[Thanks to Linda Donovan for the suggestions.]

Some Things Useful—con't.

represented, i.e., converted, as a string. HyperTalk uses the current value of the numberFormat property to format the string representation.

• When HyperCard performs an operation that requires strings as operators, such as concatenation, it will first represent any numbers as strings using numberFormat. The resulting string will remain a string even if it is put into a closed container.

• Constants, such as "pi" or "zero", represent strings of digits and are not interpreted as numbers unless forced to by an operation. Therefore, "put pi" results in "3.141592653..." but "put pi + 0" results in "3.141593" (the "+" forced "pi" to be converted to a number, which had to be converted back to a string since the message box is an open container—numberFormat was used to round to six fractional decimal places).

• Comparison is an operation that will try to interpret strings as numbers. Therefore, "3" > "20" is false, but "*3" > "*20" is true: If the goal is a string comparison and the values could be interpreted as numbers, then a non-numeric character must be concatenated to both values, such as 'if "*" & variable1 > "*" & variable2 then...' to prevent interpretation as a number. Remember that equality comparisons, "=" and "<>", use only primary ordering of characters, so case and diacriticals are ignored. Magnitude comparisons use both primary and secondary ordering.

MERRY CHRISTMAS!

From the Apple HyperCard User Group

Editorial Staff

Editor: David Leffler
Publication: Sandy Tompkins

MAKE A DIFFERENCE IN THE WORLD

Tell HyperCard

Date: _____ Phone: _____

Name: _____

MailStop or Address: _____

Versions of:

a. HyperCard _____

b. Associated software _____

_____

c. System _____

d. Finder _____

e. ImageWriter _____

f. LaserWriter _____

g. Any others _____

_____

Type of Macintosh: _____

Peripherals: _____

SEND TO:

AHUG
c/o David Leffler
Apple Computer, Inc.
MS/27-AQ
20525 Mariani Ave.
Cupertino, CA 95014

or
copy the format and
Apple-Link™ it to
HYPERBUG$

Description of problem, suggestions, or comments:

_____

_____

_____

_____

_____

_____

please write on the back if necessary